ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА»

(СПбГУТ)

Факультет инфокоммуникационных Сетей и систем (иксс)

кафедра программной инженерии и вычислительной техники (пи и вт)

Дисциплина: «Алгоритмы и структуры данных»

Проект по Теме: «Анализ временных рядов»

Выполнили:

Студенты группы ИКПИ-05

Принял:

Доцент кафедры ПИиВТ

Кузнецов М.С. Романенко К.А.

Подпись\_\_\_\_\_\_\_\_\_\_\_\_

Дагаев А.В.

Подпись\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_\_\_»\_\_\_\_\_\_\_\_2022

СОДЕРЖАНИЕ

Оглавление

[Глава 1. Построение теоретической базы 3](#_Toc104994416)

[1.1. Цель работы 3](#_Toc104994417)

[1.2. Описание работы 3](#_Toc104994418)

[Глава 2. Описание и анализ моделей и методов прогнозирования 3](#_Toc104994419)

[2.1. Прогнозирование 3](#_Toc104994420)

[2.2. Исходные данные 3](#_Toc104994421)

[2.3. Характеристики временных рядов 4](#_Toc104994422)

[2.4. Используемые регрессионные модели 4](#_Toc104994423)

[2.4.1 Регрессионный анализ 4](#_Toc104994424)

[2.4.2. LinearRegression 4](#_Toc104994425)

[2.4.2. RandomForestRegressor 6](#_Toc104994426)

[2.4.7. CatBoostRegressor 6](#_Toc104994427)

[2.5 Используемые методы преобразования данных 7](#_Toc104994428)

[2.5.1 Преобразования Бокса-Кокса 7](#_Toc104994429)

[2.5.2 Нормализация 7](#_Toc104994430)

[2.5.3 Дифференцирование 8](#_Toc104994431)

[Глава 3. Работа используемых регрессионных моделей 9](#_Toc104994432)

[3.1. Линейная регрессия 9](#_Toc104994433)

[3.2. RandomForestRegressor 10](#_Toc104994434)

[3.3. CatBoostRegressor 11](#_Toc104994435)

[3.4. Описание метрик 13](#_Toc104994436)

[3.4.1 *Средняя квадратическая ошибка – MSE* 13](#_Toc104994437)

[3.4.2 *Коэффициент детерминации - R2* 13](#_Toc104994438)

[3.4.3 *Средняя абсолютная ошибка - MAE* 14](#_Toc104994439)

[3.4.4 *Средняя абсолютная ошибка в процентах - MAPE* 14](#_Toc104994440)

[3.4.5 Зависимость точности предсказания от количества взятых предыдущих дней 15](#_Toc104994441)

[3.5. Эффективность моделей 16](#_Toc104994442)

[Глава 4. Разработка приложения 17](#_Toc104994443)

[4.1. Схемы взаимосвязей программы 17](#_Toc104994444)

[4.2. Описание библиотек и их методов 17](#_Toc104994445)

[4.3. Гиперпараметры моделей 18](#_Toc104994446)

[4.4. Структура скрипта 18](#_Toc104994447)

[4.4.1. Установка библиотек 18](#_Toc104994448)

[4.4.2. Импорт библиотек 18](#_Toc104994449)

[4.4.3. Импорт данных 18](#_Toc104994450)

[4.4.4. Обучение моделей 18](#_Toc104994451)

[4.4.5. Важность признаков при обучении 19](#_Toc104994452)

[4.4.6. Вывод результатов 19](#_Toc104994453)

[4.5. Инструкция работы с веб-скриптом 20](#_Toc104994454)

[4.6. Результаты и тестирование веб-скрипта 21](#_Toc104994455)

[Вывод 22](#_Toc104994456)

[Используемые источники 22](#_Toc104994457)

[Приложение 22](#_Toc104994458)

Глава 1. Построение теоретической базы

* 1. Цель работы

Цель настоящей работы составляет изучение и освоение методов машинного обучения, анализ и сравнение эффективности существующих моделей прогнозирования на открытых финансовых данных, а так же построение рабочего веб-скрипта для их прогнозирования.

* 1. Описание работы

Для достижения цели, сперва мы загрузим данные криптовалюты VeChain в формате pandas DataFrame с помощью библиотеки yfinance.

Обработаем их с помощью методов дифференцирования, преобразования Бокса-Кокса и нормализации с помощью разработанных функций на основе библиотеки sklearn и pandas со встроенными для этого методами. Затем обучим на разных данных с помощью встроенных в библиотеки sklearn и catboost методов несколько моделей машинного обучения, таких как линейная регрессия, случайный лес и градиентный бустинг.

Сравним результаты с помощью метрик MSE, MAE, MAPE и R2, взятых из библиотеки sklearn.metrics. После сравнения и графического анализа полученных результатов поймем на каких данных и какие модели дают наилучшие результаты. В завершении построим на этой основе веб-скрипт для предсказания коридора, то есть наивысшей и наинизшей цен для следующего промежутка времени и разместим его на платформе Google Colab, для удаленного доступа к написанному нами веб-скрипту с любого устройства.

Глава 2. Описание и анализ моделей и методов прогнозирования

2.1. Прогнозирование

Прогнозирование — это задача науки о данных, которая является центральной для многих видов деятельности в организации. Например, крупные организации должны участвовать в планировании мощностей, чтобы эффективно распределять ограниченные ресурсы и ставить цели для измерения производительности относительно базового уровня.

Создание высококачественных прогнозов - непростая задача ни для машин, ни для большинства аналитиков. Полностью автоматические методы прогнозирования могут быть хрупкими и часто слишком негибкими, а аналитики, которые могут составлять высококачественные прогнозы, встречаются довольно редко, потому что прогнозирование — это специализированный навык науки о данных, требующий значительного опыта.

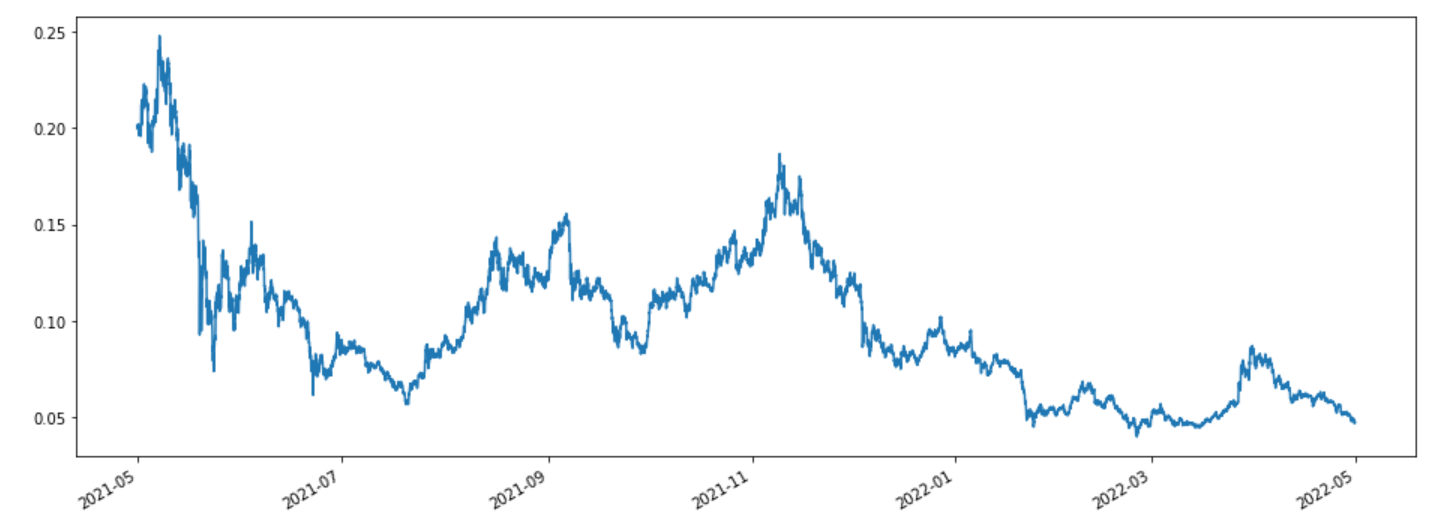
2.2. Исходные данные

За исходные данные мы берем стоимость пары криптовалюты VeChain к доллару США или «VET-USD» в период с 1 мая 2021 года по 1 мая 2022 года с шагом в один час.

Нами была выбрана данная криптовалюта из-за её достоточной волатильности, кореляции с Etherium и Bitcoin, а так же достаточной популярности среди других криптовалют (36 место по капитализации среди криптовалют), что обеспечивает достаточное колличество сделок для анализа.

Данные брались с источника «Yahoo! Finance» - одного из главных поставщиков подобной информации о стоимости акций и криптовалют.

К исходным данным относятся данные открытия, закрытия, максимальная и минимальная цены криптовалюты, объемы торгов на каждом из промежутков времени.

Рис 1. График изменения стоимости криптовалюты VeChain

2.3. Характеристики временных рядов

Модели, которые построены по данным, характеризующим один объект за ряд определенных последовательных периодов, называется моделями временных рядов. Существует множество моделей для предсказания временных рядов, мы рассматриваем линейную регрессию, случайный лес и CatBoost.

Для повышения точности прогнозов будем изменять начальные данные с помощью дифференирования, нормализации и преобразования Бокса-Кокса.

2.4. Используемые регрессионные модели

2.4.1 Регрессионный анализ

Регрессионный анализ — это набор статистических методов оценки отношений между переменными. Его можно использовать для оценки степени взаимосвязи между переменными и для моделирования будущей зависимости. По сути, регрессионные методы показывают, как по изменениям «независимых переменных» можно зафиксировать изменение «зависимой переменной».

Зависимую переменную в бизнесе называют предиктором (характеристика, за изменением которой наблюдают). Это может быть уровень продаж, риски, ценообразование, производительность и так далее. Независимые переменные — те, которые могут объяснять поведение выше приведенных факторов (время года, покупательная способность населения, место продаж и многое другое). Регрессионный анализ включает несколько моделей.

2.4.2. LinearRegression

LinearRegression - это математическая модель, которая описывает связь нескольких переменных. Модели линейной регрессии представляют собой статистическую процедуру, помогающую прогнозировать будущее. Она применяется в научных сферах и в бизнесе, а в последние десятилетия используется в машинном обучении

 y=ax+b

. Задача регрессии в машинном обучении — предсказание одного параметра (y) по известному параметру x, где x — набор параметров, характеризующий наблюдение, и с помощью коэффицентов a, b.

2.4.2. RandomForestRegressor

Random forest— это множество решающих деревьев. В задаче регрессии их ответы усредняются, в задаче классификации принимается решение голосованием по большинству.

Для построения каждого расщепления в дереве просматриваем max\_features случайных признаков (для каждого нового расщепления — свои случайные признаки). Выбираем наилучшие признак и расщепление по нему (по заранее заданному критерию). Дерево строится, как правило, до исчерпания выборки (пока в листьях не останутся представители только одного класса), но в современных реализациях есть параметры, которые ограничивают высоту дерева, число объектов в листьях и число объектов в «подвыборке», при котором проводится расщепление. Понятно, что такая схема построения соответствует главному принципу «ансамблирования» (построению алгоритма машинного обучения на базе нескольких, в данном случае решающих деревьев): базовые алгоритмы должны быть хорошими и разнообразными (поэтому каждое дерево строится на своей обучающей выборке и при выборе расщеплений есть элемент случайности).

2.4.7. CatBoostRegressor

Практически любой современный метод на основе градиентного «бустинга» работает с числовыми признаками. Если у нас в наборе данных присутствуют не только числовые, но и категориальные признаки, то необходимо переводить категориальные признаки в числовые. Это приводит к искажению их сути и потенциальному снижению точности работы модели. Именно поэтому было важно разработать алгоритм, который умеет работать не только с числовыми признаками, но и с категориальными напрямую, закономерности между которыми этот алгоритм будет выявлять самостоятельно, без ручной «помощи». CatBoost — библиотека для градиентного «бустинга», главным преимуществом которой является то, что она одинаково хорошо работает «из коробки» как с числовыми признаками, так и с категориальными.

Алгоритм работы, следующий: для каждого документа имеется набор значений признаков, имеется дерево, в вершинах дерева — условия. Если условие выполнено, осуществляется переход в правую ветвь вершины, иначе в левого. Нужно пройти до листа по дереву в соответствии со значениями признаков для документа. На выходе каждому документу соответствует значение листа. Это и есть ответ.

Идея «бустинг» - подхода заключается в комбинации слабых (с невысокой обобщающей способностью) функций, которые строятся в ходе итеративного процесса, где на каждом шаге новая модель обучается с использованием данных об ошибках предыдущих. Результирующая функция представляет собой линейную комбинацию базовых, слабых моделей.

Итого при достаточно большом количестве деревьев можно сильно уменьшить ошибку, однако не стоит забывать, что чем больше деревьев, тем дольше обучается модель и в какой-то момент прирост качества становится незначительным.

В основе CatBoost лежит градиентный «бустинг».

* Градиент функции ошибки — все производные по всем значениям функции
* Градиентный «бустинг» — метод машинного обучения, который создает решающую модель прогнозирования в виде ансамбля слабых моделей прогнозирования, обычно деревьев решений. Он строит модель поэтапно, позволяя оптимизировать произвольную дифференцируемую функцию потерь.

2.5 Используемые методы преобразования данных

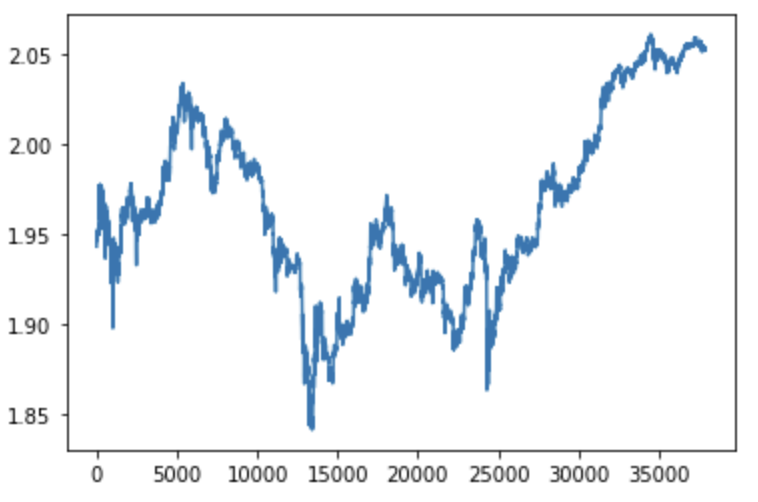
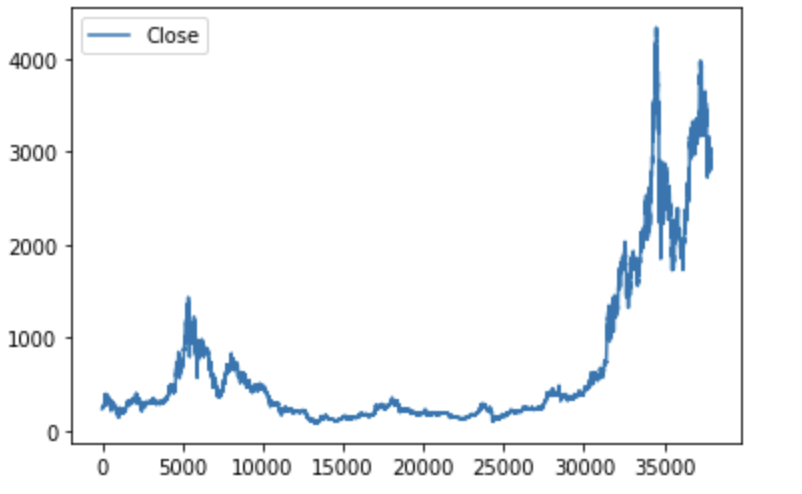
2.5.1 Преобразования Бокса-Кокса

Одной из задач Бокс-Кокс преобразования является приведение закона распределения входной последовательности к "нормальному" виду, что особенно полезно в моделях линейной регрессии.

Для исходной последовательности  однопараметрическое преобразование Бокса-Кокса с параметром \lambda определяется следующим образом:



Параметр \lambda можно выбирать, максимизируя логарифм правдоподобия. Еще один способ поиска оптимального значения параметра основан на поиске максимальной величины коэффициента корреляции между квантилями функции нормального распределения и отсортированной преобразованной последовательностью.



*Рис.2 – Результаты до и после преобразования данных методом Бокса-Кокса*

2.5.2 Нормализация

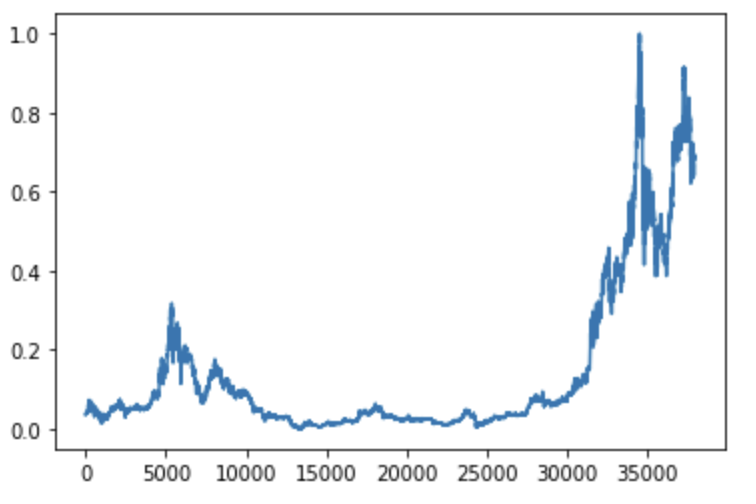
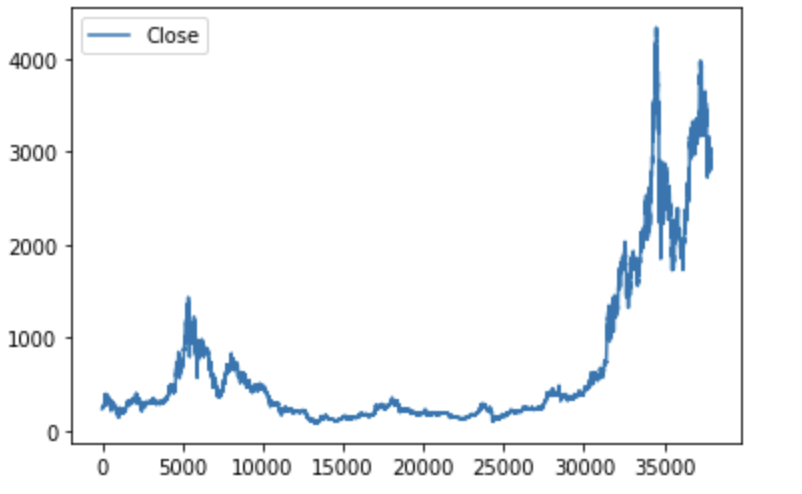
Ключевая цель нормализации — приведение различных данных в самых разных единицах измерения и диапазонах значений к единому виду, который позволит сравнивать их между собой или использовать для расчёта схожести объектов. На практике это необходимо, например, для кластеризации и в некоторых алгоритмах машинного обучения.

– значение последовательности

– минимальное значение последовательности

– максимальное значение последовательности

В нашем случае нормализация может быть полезна только для линейной регрессии, так как для бустингов и деревьев, ввиду особенности их работы, ничего не поменяется.



*Рис.3 – Результаты до и после нормализации данных*

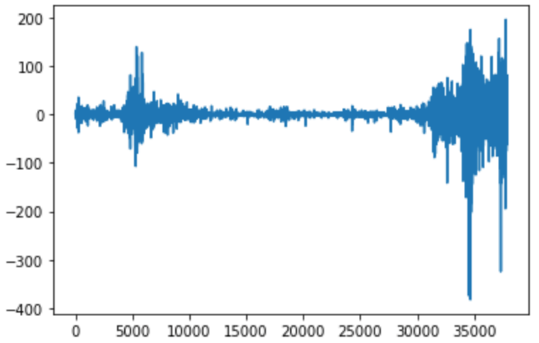
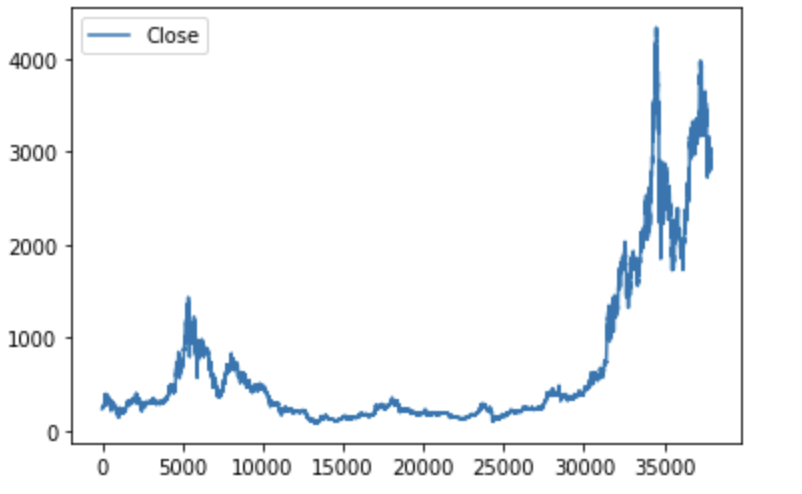
2.5.3 Дифференцирование

Дифференцированием временного ряда называют вычисление разницы следующего члена временного ряда от предыдущего.

y(t+1) – следующий член временного ряда

y(t) –предыдущий член временного ряда

Для деревьев и бустингов крайне важно, чтобы анализируемый ряд не имел «тренда», то есть был стационарным. Дифференцирование прекрасно справляется с подобной задачей, преобразуя данные между соседними промежутками времени в разницу между ними.



*Рис.4 – Результаты до и после дифференцирования данных*

Глава 3. Работа используемых регрессионных моделей

Ниже представлены графики реальной (синий) и предсказанной (оранжевый) цен по часовым свечкам для описанных нами моделей регрессии. Все модели предсказывают каждую точку по 10 стоящим перед ней. Каждая модель работает с исходными и дифференцированными данными, а так же дополнительными данными зависимости от BitCoin.

3.1. Линейная регрессия

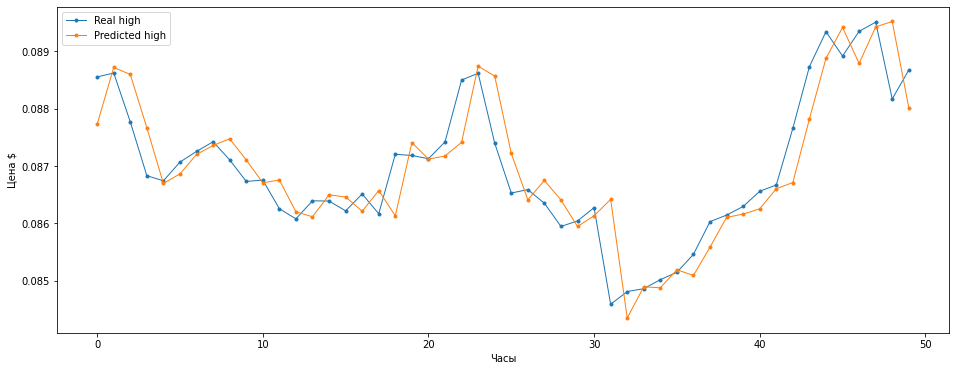


Рис. 5. График предсказания цены моделью линейной регрессии

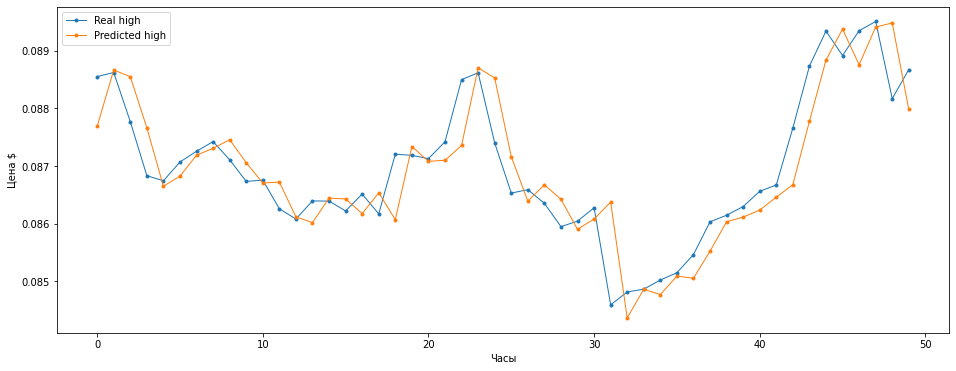


Рис. 6. График предсказания цены моделью линейной регрессии с использованием дифференцирования

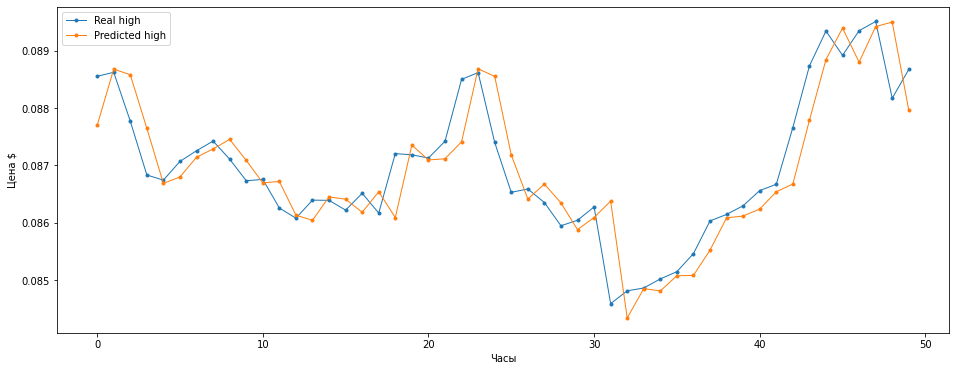


Рис. 7. График предсказания цены моделью линейной регрессии без использования дифференцирования и с использованием преобразования Бокса-Кокса

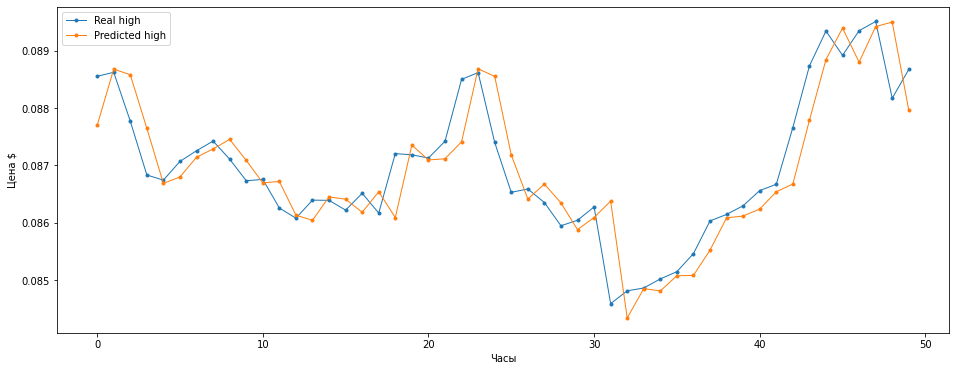


Рис. 8. График предсказания цены моделью линейной регрессии без использования дифференцирования и с использованием преобразования Бокса-Кокса и нормализации

3.2. RandomForestRegressor

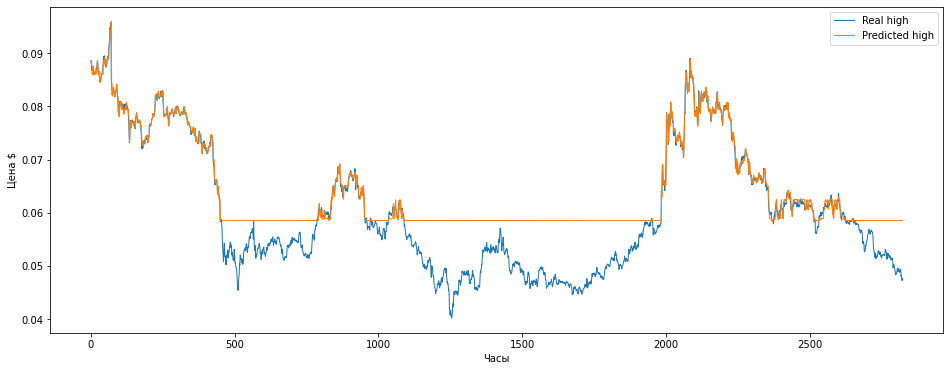


Рис. 9. График предсказания цены моделью случайного леса

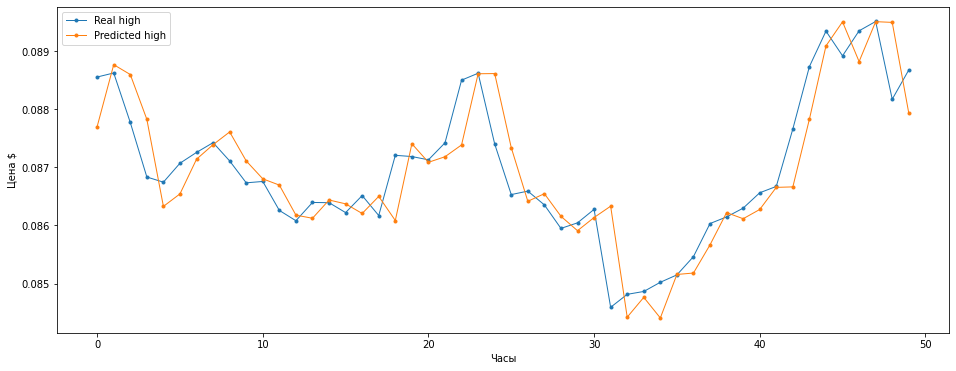


Рис. 10. График предсказания цены моделью случайного леса с использованием дифференцирования

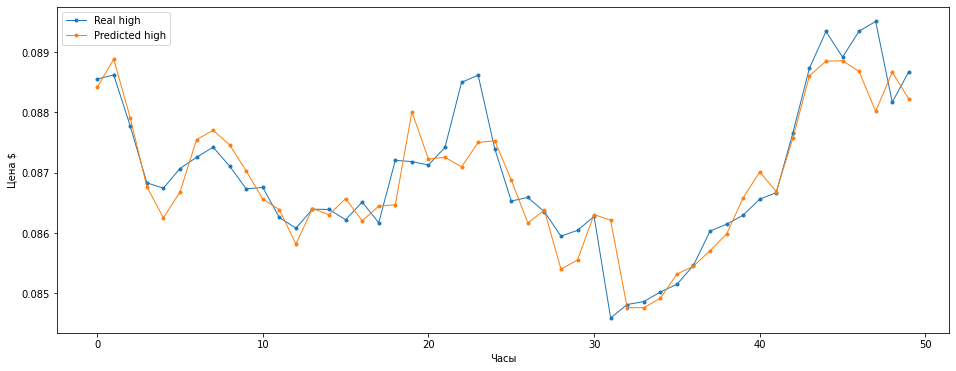
****

Рис. 11. График предсказания цены моделью случайного леса с использованием дифференцирования и учетом зависимости от Bitcoin

3.3. CatBoostRegressor

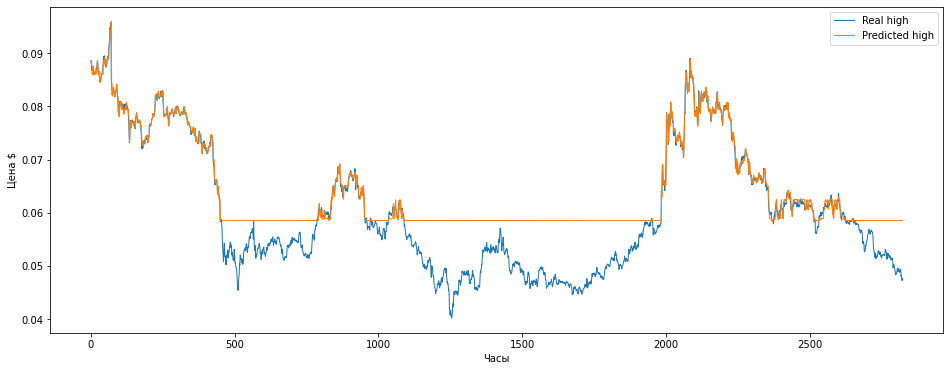


Рис. 12. График предсказания цены акции моделью градиентного бустинга

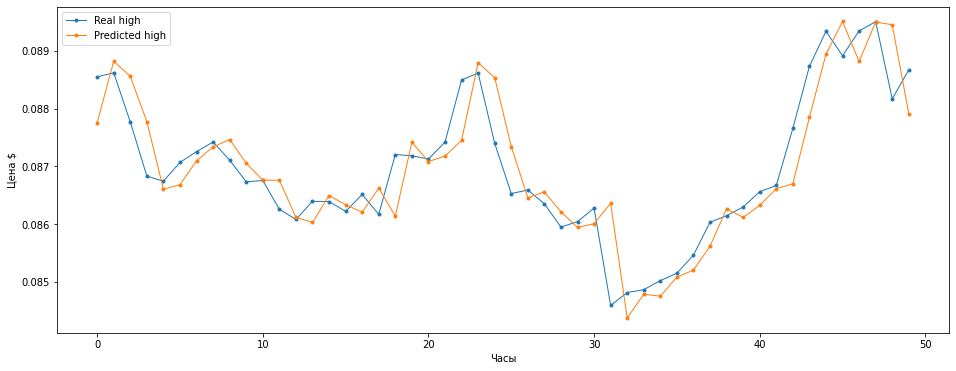


Рис. 13. График предсказания цены моделью градиентного бустинга с использованием дифференцирования

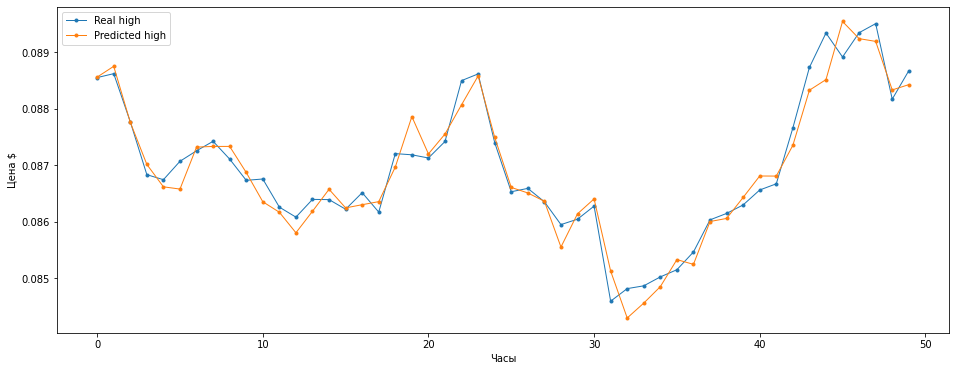
****

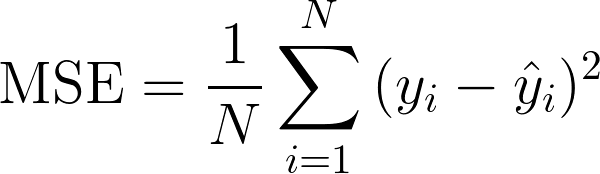
Рис. 14. График предсказания цены моделью градиентного бустинга с использованием дифференцирования и учетом зависимости от Bitcoin

Из графиков ясно, что нелинейные модели не могут работать на исходных, нестационарных данных. В свою очередь линейные модели не могут работать с большими объемами данных, из-за чего с их увеличением, эффективность линейной регресии снижается.

## 3.4. Описание метрик

### 3.4.1 *Средняя квадратическая ошибка – MSE*

MSE - среднеквадратическая ошибка. Измеряет среднее квадратов ошибок, то есть среднеквадратичную разницу между оценочными и фактическими значением. Чем эта разница меньше, тем лучше модель предсказывает значения.



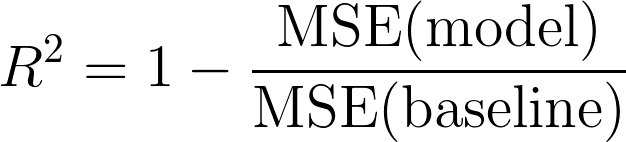
Где N – колличество предсказанных данных

yi - реальные данные (фактические значения)

ŷi - предсказанные данные (оценочные значения)

### 3.4.2 *Коэффициент детерминации - R2*

R-квадрат – это коэффициент линейной детерминации. Показывает, во сколько предсказанные нашей моделью данные лучше, чем среднее значение. Чем ближе к 1, тем лучше работает модель.

****

Где MSE(model) – среднеквадратичное отклонение между фактическими и предсказанными данными

MSE(baseline) - среднеквадратичное отклонение между фактическими данными и средним значением

Характеризует степень сходства исходных данных и предсказанных. В отличии от MSE не зависит от единиц измерения данных, поэтому поддается сравнению

Когда R² отрицательно, это означает, что модель хуже, чем предсказание среднего значения

### 3.4.3 *Средняя абсолютная ошибка - MAE*

MAE – средняя абсолютная ошибка. Измеряет среднеабсолютную разницу между оценочными и фактическими значением. Чем эта разница меньше, тем лучше модель предсказывает значения.

Где N – колличество предсказанных данных

yi - реальные данные (фактические значения)

ŷi - предсказанные данные (оценочные значения)

### 3.4.4 *Средняя абсолютная ошибка в процентах - MAPE*

MAPE - средняя абсолютная процентная ошибка. Измеряет среднее абсолютное процентное отклонение между оценочными и фактическими значением, является мерой точности прогноза. Чем меньше, тем лучше модель предсказывает значения.

Где N – колличество предсказанных данных

yi - реальные данные (фактические значения)

ŷi - предсказанные данные (оценочные значения)

MAPE можно измерять в долях или процентах. Если он например равен 0.11, то это говорит о том, что ошибка составила 11 % от фактических значений.

3.4.5 Зависимость точности предсказания от количества взятых предыдущих дней

Ниже представлены графики зависимости работы модели CatBoost от колличества лаговых фичей (колличества взятых предыдущих точек) для предсказания одной следующей по метрикам MSE, MAPE и R2.

Из графиков видно, что при увеличении количества взятых предыдущих точек для предсказания следующей, модели работают лучше по всем метрикам.

## 3.5. Эффективность моделей

|  |  |  |  |
| --- | --- | --- | --- |
| **Без преобразований** | | | |
|  | **LinearRegression** | **RandomForestRegressor** | **CatBoostRegressor** |
| **MSE** | 2,44E-07 | 1,14E-05 | 2,16E-06 |
| **MAE** | 0,000341439 | 0,003032631 | 0,001116104 |
| **MAPE** | 0,00561768 | 0,047441466 | 0,018375411 |
| **R2** | 0,886587309 | -2,045632 | -2,119898 |
| **Дифференцирование** | | | |
| **MSE** | 2,46E-07 | 2,49E-07 | 2,60E-07 |
| **MAE** | 0,00034473 | 0,000347381 | 0,000354213 |
| **MAPE** | 0,00567991 | 0,005728233 | 0,005831853 |
| **R2** | 0,89239508 | 0,889871705 | 0,877883247 |
| **Дифференцирование + BTC** | | | |
| **MSE** | 7,39E-07 | 9,07E-08 | 8,04E-08 |
| **MAE** | 0,000689551 | 0,000201506 | 0,000206301 |
| **MAPE** | 0,01155619 | 0,003332528 | 0,003406357 |
| **R2** | 0,78427254 | 0,961175389 | 0,965539517 |
| **Дифференцирование + Бокс-Кокс + BTC** | | | |
| **MSE** | 5,36E-07 | 9,36E-08 | 9,34E-08 |
| **MAE** | 0,000632356 | 0,000207116 | 0,000218858 |
| **MAPE** | 0,010467014 | 0,003418947 | 0,003610056 |
| **R2** | 0,803756025 | 0,958533815 | 0,959471415 |

Таблица 1. Таблица метрик

В таблице представлены результаты работы моделей по метрикам MSE, MAE, MAPE и R2.

На данных без преобразований лучше всего себя показывает линейная регрессия, так как для ее работы не нужно большое количество стационарных данных. В свою очередь из-за этого случайный лес и градиентный бустинг показывают плохой результат.

Соответственно благодаря дифференцированию случайный лес и градиентный бустинг приближаются к значения линейной регрессии. Однако при увеличении колличества данных, линейная регрессия уже не справляется с предсказанием также хорошо, как нелинейные модели.

И наконец при добавлении к изначальным дифференцированию и зависимостям от BitCoin преобразования Бокса-Кокса, данные только стали хуже, так как из-за большого колличества преобразований, моделям стало сложнее предсказывать временной ряд.

Глава 4. Разработка приложения

4.1. Схемы взаимосвязей программы

Скрипт написан на языке Python в онлайн среде Google Colab с использованием различных библиотек.

4.2. Описание библиотек и их методов

1. sklearn – самая распространённая библиотека для машинного обучения.

* metrics – модуль со стандартными метриками.
* r2\_score – функция для вычисления коэффициента детерминации
* mean\_squared\_error – функция для вычисления средней квадратичной ошибки
* linear\_model – модуль содержащий набор линейный моделей
* LinearRegression – классическая модель линейной регрессии
* ensemble – модуль содержащий методы основанные на идеи ансамблей
  + RandomForestRegressor – регрессия Random Forest
* train\_test\_split – модуль для разбивки исходных данных для обучения и тестирования
* fit – метод для обучения модели

1. catboost – библиотека содержащая в себе алгоритмы Cat Boost

* CatBoostRegressor – регрессия Cat Boost
* get\_feature\_importance – получение списка используемых метрик, отсортированных по важности для предсказаний
* fit – метод для обучения модели

1. yfinance – API для загрузки данных об акции с акции.

* Ticker – модуль для получения данных по тикеру
* history – метод позволяющий получить данные об акциях компании за выбранный период

1. pandas – библиотека для обработки и анализа данных
   * iloc – обращение к данным DataFrame по индексу
   * DataFrame – табличная структура данных
2. numpy – библиотека для работы с массивами
   * diff – метод для дифференцирования данных
3. matplotlib – библиотека для визуализации данных
   * plot – метод для построения графиков
4. Project.ipynb – основная программа
5. ta – библиотека с техническими индикаторами
   * add\_all\_ta\_features – добавление всех доступных в библиотеке ta технических индикаторов
6. scipy – библиотека для преобразования данных
   * boxcox – метод для преобразования данных методом Бокса-Кокса
   * inv\_boxcox - метод для обратного преобразования данных методом Бокса-Кокса

4.3. Гиперпараметры моделей

LinearRegression() – значения по умолчанию:

* fit\_intercept=True – преобразование данных перед обучением
* normalize='deprecated' – нормализация данных
* copy\_X=True – параметр Х не будет перезаписываться
* n\_jobs=None – использование всех ядер процессора
* positive=False – элементы могут быть отрицательными

RandomForestRegressor(max\_depth=100, random\_state=0, min\_samples\_leaf=10):

* max\_depth=100 – максимальная глубина дерева
* random\_state=0 – исключение возможности перемешивания данных
* min\_samples\_leaf=10 – минимальное колличество путей к каждому листу

CatBoostRegressor(iterations = 500, learning\_rate = 0.03, depth = 8, l2\_leaf\_reg = 7):

* iterations = 500 – колличество итераций
* learning\_rate = 0.03 - размер шага на каждой итерации
* depth = 8 – глубина обучения
* l2\_leaf\_reg = 7 – колличество листьев

4.4. Структура скрипта

4.4.1. Установка библиотек

Уставка библиотек осуществляется с помощью команды

!pip install <название библиотеки>, например:

!pip install joblib

!pip install ta

!pip install yfinance

!pip install talib-binary

!pip install catboost

4.4.2. Импорт библиотек

Уставка библиотек осуществляется с помощью команды import, например:

import numpy as np

import pandas as pd

4.4.3. Импорт данных

Импорт данных осуществляется с помощью команд tickerData.history библиотеки от yahoo finance

tickerData = yf.Ticker(tickerS)

tickerDf = tickerData.history(period = period\_, interval = interval\_)

4.4.4. Обучение моделей

Обучение моделей происходит с помощью метода fit

regressor.fit(X,  y)

4.4.5. Важность признаков при обучении

Важность признаков расчитывается с помощью команды get\_feature\_importance

regressor.get\_feature\_importance()

4.4.6. Вывод результатов

Вывод результатов происходит с помощью методов plot и print

df\_for = pd.DataFrame([df1\_high['High'], df1\_high['Predict\_high'],

                       df1\_low['Low'], df1\_low['Predict\_low']]).T

ax = df\_for[-vivod-2:].plot(figsize=(16,6), linewidth = 1, style='.-',

                            ylabel = 'Цена $', xlabel = 'Часы',

                            xticks = range(-vivod,2,1))

ax.set\_xticklabels(list(range(-vivod,0,1))+['last', 'next'])

print('High:', df1\_high['High'].iloc[-2],' -> ',

      df1\_high['Predict\_high'].iloc[-1],

      '\nLow: ', df1\_low['Low'].iloc[-2], ' -> ',

      df1\_low['Predict\_low'].iloc[-1])

4.5. Инструкция работы с веб-скриптом

Шаг 1. Открываем скрипт по ссылке https://colab.research.google.com/drive/1vQW4WqqpK13vLnZShP2gAIIBhCYtKuZc?usp=sharing

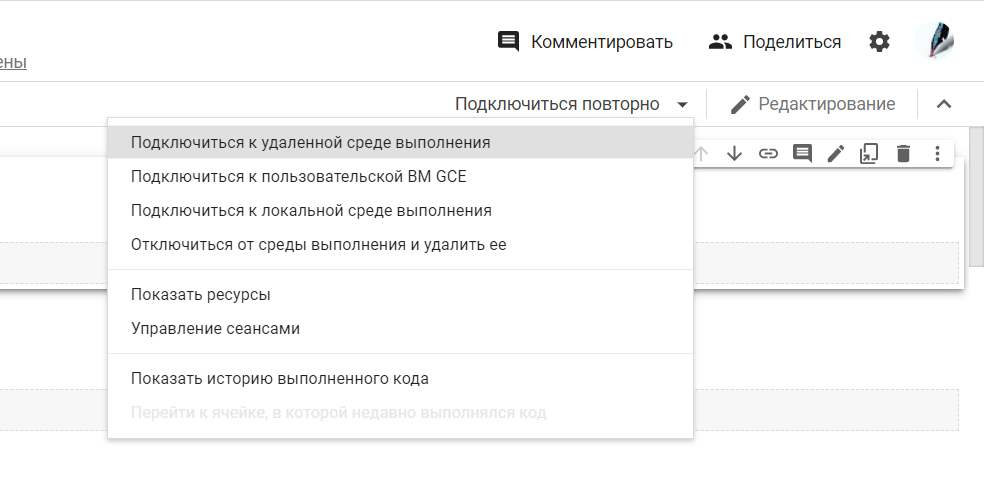


Рис. 15. Подключение к удаленной среде выполнения

Шаг 2. Нажимаем кнопку «Подключиться к удаленной среде выполнения» (рис. 12).

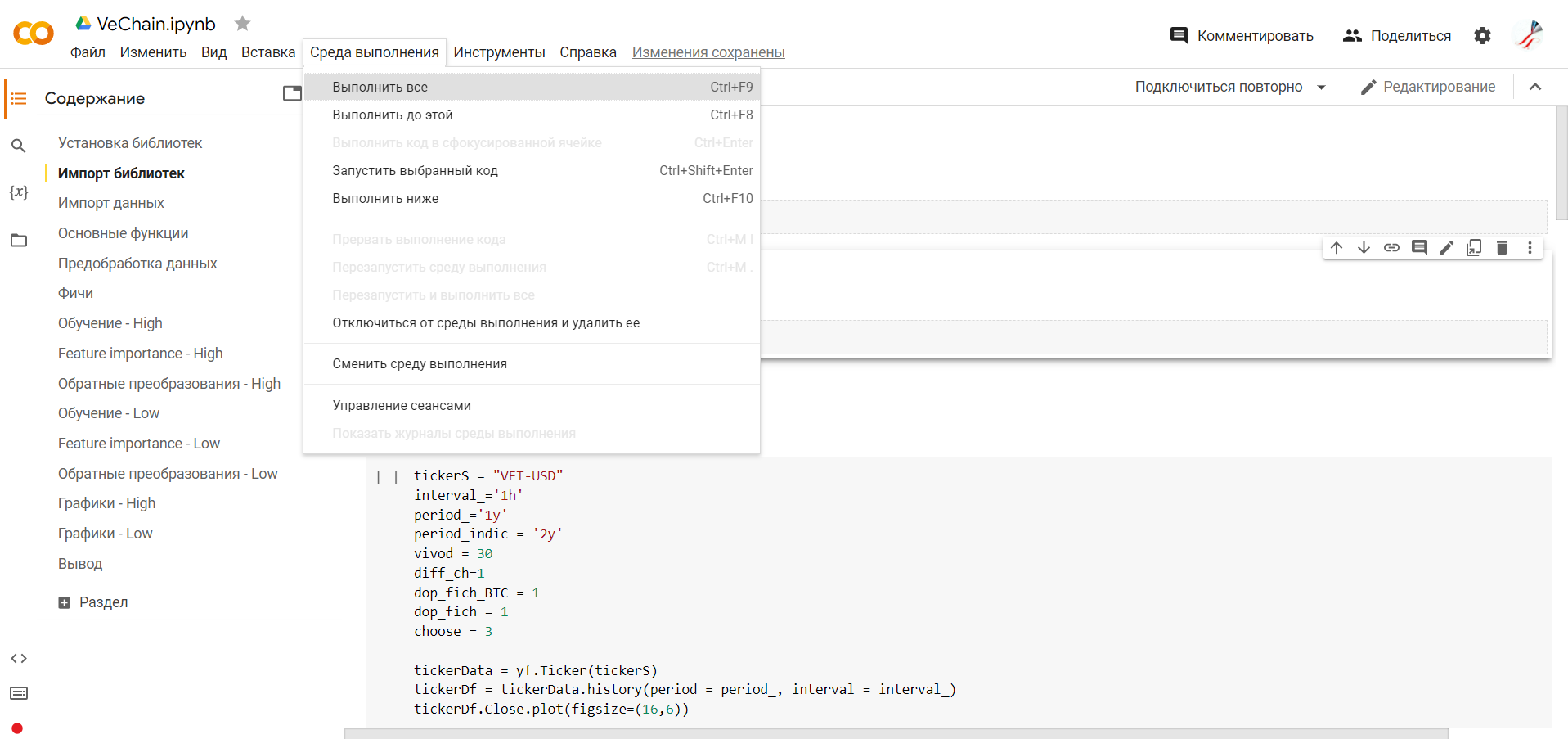


Рис. 16. Запуск скрипта

Шаг 3. Нажимаем кнопку «Выпонить все» в «Среда выполнения» (рис. 13). И ждем окончания работы скрипта.

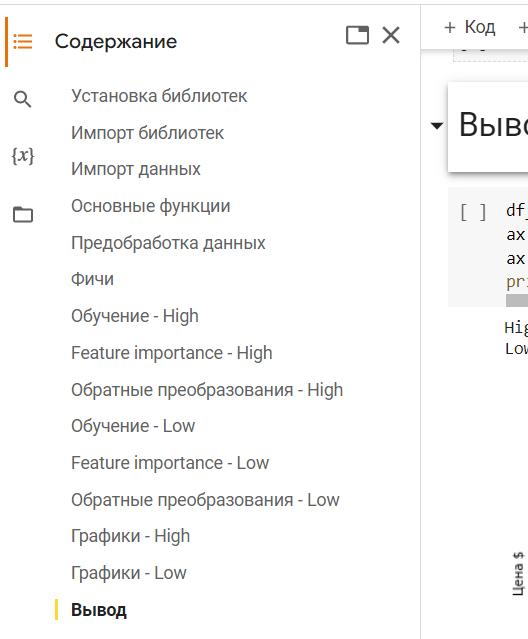


Рис. 17. Переход к выводу

Шаг 4. В содержании, нажимаем на «Вывод» (рис. 14) или просто листаем до конца страницы вниз.

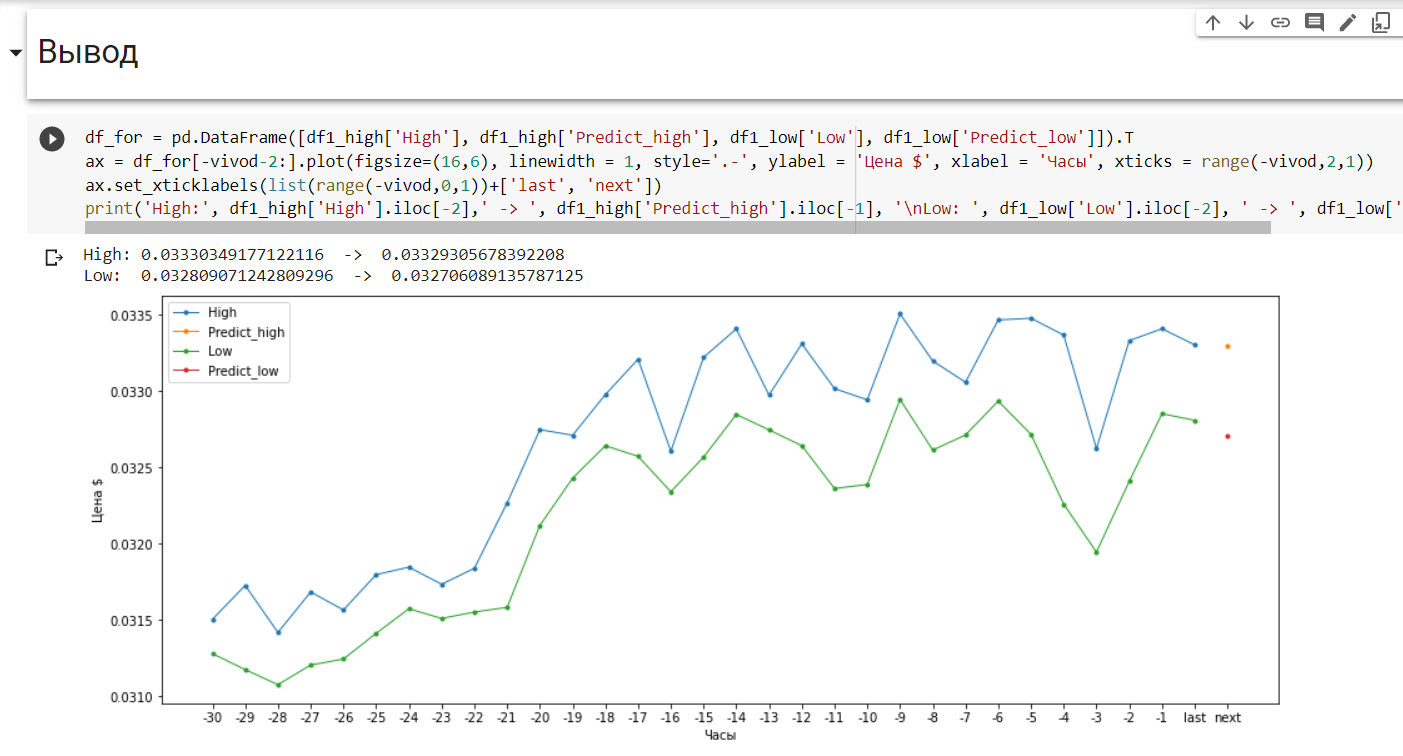


Рис. 18. Вывод

Шаг 5. Наблюдаем за динамикой развития цен на последних 30 часовых свечках(рис. 15). Где синим и зеленым цветами указаны максимальная и минимальная цены промежутка соответсвенно, а оранжевая и красная точки показывают вероятные изменения данного диапозона цен.

Так же видим предсказанные значения коридора цены High и Low в конкретных значениях над графиком (обведены красным). Эти данные показывают примерное движение цены в будущем и позволяет сделать выводы о дальнейшей торговле.

4.6. Результаты и тестирование веб-скрипта

Функции веб-скрипта работают в соответствии с разработанной логикой, ошибок в работе веб-скрипта не наблюдается, данные отображаются корректно.

Вывод

В ходе выполнения проектной работы были изучены и освоены методы машинного обучения, проведено сравнение эффективности существующих моделей прогнозирования на открытых финансовых данных криптовалюты VeChain.

* 1. Были рассмотрены ключевые понятия, связанные с машинным обучением.

1. Решена задача регрессии, проанализированы статистические данные, представлен график изменения курса криптовалюты. Выполнен анализ регрессионных моделей, дано их описание.
2. Примененен ряд моделей регрессии, RandomForestRegressor, CatBoostRegressor на языке программирования Python. Проведено сравнение эффективности данных моделей с помощью метрики mape, mae, mse и R2. По полученным результатам можно сделать вывод, что преобразования данных не всегда необходимы для работы моделей, таких как линейная регрессия, которая показала лучший результат по метрикам именно без предварительных преобразований. Однако остро необходимы для ветвящихся моделей, таких как случайный лес и CatBoost, которые показали лучшие результаты именно с преобразованиями.
3. Разработан веб-скрипт на платформе Google Colab, где можно в реальном времени предсказать изменения курса криптовалюты VeChain.

Используемые источники

1. <https://scikit-learn.org/stable/> - сайт описывающий работу библиотеки sclearn
2. <https://catboost.ai/> - сайт описывающий работу библиотеки catboost
3. Книга: «Фрактальный анализ валютных временных рядов» Цветков В.П., Цветков И.В., Гуляева О.С.
4. Книга: «Нелинейная динамика и анализ временных рядов» Котляров О.Л., Лоскутов А.Ю.
5. Книга: «Взвешенный прогноз на основе анализа временных рядов» Кузнецов А.А. Журов А.В.

Приложение

#%% md

# Установка библиотек

#%%

!pip install joblib

!pip install ta

!pip install yfinance

!pip install talib-binary

!pip install catboost

#%% md

# Импорт библиотек

#%%

import numpy as np

import pandas as pd

import joblib

import talib

import catboost

import scipy

import sklearn

from ta import add\_all\_ta\_features

import yfinance as yf

from scipy.stats import boxcox

from scipy.special import inv\_boxcox

from sklearn.preprocessing import MinMaxScaler

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

from sklearn.ensemble import RandomForestRegressor

from catboost import CatBoostRegressor

#%% md

# Импорт данных

#%%

tickerS = "VET-USD"

interval\_='1h'

period\_='1y'

period\_indic = '2y'

vivod = 30

diff\_ch=1

dop\_fich\_BTC = 1

dop\_fich = 1

choose = 3

tickerData = yf.Ticker(tickerS)

tickerDf = tickerData.history(period = period\_, interval = interval\_)

tickerDf.Close.plot(figsize=(16,6))

#%% md

# Основные функции

#%%

def box\_cox(data, columnOld, columnNew):

trans, lambda\_ = boxcox(data[columnOld].values)

data[columnNew] = trans

return lambda\_

def inv\_box\_cox(data, lambda\_):

return inv\_boxcox(data, lambda\_)

def diff(data, columnOld, columnNew):

data[columnNew] = data[columnOld].diff()

def inv\_diff(base, predict):

summ=[i+j for i,j in zip(base[len(base)-len(predict)-1:-1], predict)]

return np.array(summ).reshape(-1,1)

def minmax(data, columnOld, columnNew):

scaler = MinMaxScaler()

data[columnNew] = scaler.fit\_transform(data[columnOld].values.reshape(-1,1))

return scaler.fit(data[columnOld].values.reshape(-1,1))

def inv\_minmax(data, obj):

return obj.inverse\_transform(data).reshape(-1,1)

def lag\_fich(data, columnOld):

prediction\_window = 10

prediction\_columns = [columnOld]

for i in range(1, prediction\_window+1):

col\_name = f'shift\_{i}'

prediction\_columns.append(col\_name)

data[col\_name] = data[prediction\_columns[0]].shift(i)

return data[prediction\_columns].dropna(), prediction\_columns

def all\_transform(dataf, pupu, box\_cox\_ch=1):

if box\_cox\_ch == 1:

lambda\_dataf = box\_cox(dataf, pupu, 'boxcox')

data1 = dataf['boxcox'].copy()

diff(dataf, 'boxcox', 'diff')

return lambda\_dataf, data1

else:

lambda\_dataf = 0

data1 = dataf[pupu].copy()

diff(dataf, pupu, 'diff')

return lambda\_dataf, data1

box\_cox\_ch=0

def inv\_all\_transform(data1, data\_test, lambda\_data, box\_cox\_ch=1):

if box\_cox\_ch == 1:

inv\_data\_test\_from\_diff\_to\_boxcox = inv\_diff(data1, data\_test.values.reshape(-1,1))

inv\_data\_test = inv\_boxcox(inv\_data\_test\_from\_diff\_to\_boxcox, lambda\_data)

return inv\_data\_test

else:

inv\_data\_test = inv\_diff(data1, data\_test.values.reshape(-1,1))

return inv\_data\_test

def add\_feature(name, feature, prediction\_columns, high, low):

prediction\_columns.append(name)

high[name] = feature[len(feature)-len(high):]

low[name] = feature[len(feature)-len(low):]

def metrics(predicted, real):

from sklearn.metrics import mean\_squared\_error

print('mean\_squared\_error (MSE)', mean\_squared\_error(predicted,real))

from sklearn.metrics import mean\_absolute\_error

print('mean\_absolute\_error (MAE)', mean\_absolute\_error(predicted, real))

from sklearn.metrics import mean\_absolute\_percentage\_error

print('mean\_absolute\_percentage\_error (MAPE)', mean\_absolute\_percentage\_error(predicted, real))

from sklearn.metrics import r2\_score

print('r2\_score (R^2)', r2\_score(predicted,real))

#%% md

# Предобработка данных

#%%

low = tickerDf[:].copy()

high = tickerDf[:].copy()

close = tickerDf[:].copy()

high\_real = high['High'].copy()

low\_real = low['Low'].copy()

#%%

if diff\_ch ==1:

lambda\_high, high1 = all\_transform(high, 'High', box\_cox\_ch)

lambda\_low, low1= all\_transform(low, 'Low', box\_cox\_ch)

#%%

if box\_cox\_ch == 1:

high['boxcox'].plot(figsize=(16,6), linewidth = 1)

#%%

if diff\_ch == 1:

high, prediction\_columns = lag\_fich(high, 'diff')

low, prediction\_columns = lag\_fich(low, 'diff')

else:

high, prediction\_columns\_high = lag\_fich(high, 'High')

low, prediction\_columns\_low = lag\_fich(low, 'Low')

#%%

if diff\_ch == 1:

high['diff'].plot(figsize=(16,6), linewidth = 1)

#%% md

# Фичи

#%%

if dop\_fich\_BTC == 1:

''''''

tickerBTC = "BTC-USD"

tickerDataBTC = yf.Ticker(tickerBTC)

tickerDf\_features\_BTC = tickerDataBTC.history(period = period\_, interval = interval\_)

featuresBTC\_high = tickerDf\_features\_BTC[:-1].shift().copy()

featuresBTC\_low = tickerDf\_features\_BTC[:-1].shift().copy()

\_ , \_ = all\_transform(featuresBTC\_high, 'High', box\_cox\_ch)

\_ , \_ = all\_transform(featuresBTC\_low, 'Low', box\_cox\_ch)

prediction\_columns.append('BTC')

high['BTC'] = featuresBTC\_high['diff'][len(featuresBTC\_high['diff'])-len(high):]

low['BTC'] = featuresBTC\_low['diff'][len(featuresBTC\_low['diff'])-len(low):]

tickerDf\_features\_BTC\_indic = tickerDataBTC.history(period = period\_, interval = interval\_)

features = tickerDf\_features\_BTC\_indic[:-1].copy()

output = talib.WILLR(features['High'], features['Low'], features['Close'], timeperiod=14)

add\_feature('WILLR\_BTC', output, prediction\_columns, high, low)

output = talib.PLUS\_DI(features['High'], features['Low'], features['Close'], timeperiod=14)

add\_feature('PLUS\_DI\_BTC', output, prediction\_columns, high, low)

output = talib.DX(features['High'], features['Low'], features['Close'], timeperiod=14)

add\_feature('DX\_BTC', output, prediction\_columns, high, low)

output = talib.CCI(features['High'], features['Low'], features['Close'], timeperiod=14)

add\_feature('CCI\_BTC', output, prediction\_columns, high, low)

output = talib.ADXR(features['High'], features['Low'], features['Close'], timeperiod=14)

add\_feature('ADXR\_BTC', output, prediction\_columns, high, low)

output = talib.ULTOSC(features['High'], features['Low'], features['Close'], timeperiod1=7, timeperiod2=14, timeperiod3=28)

add\_feature('ULTOSC\_BTC', output, prediction\_columns, high, low)

output = talib.ADX(features['High'], features['Low'], features['Close'], timeperiod=14)

add\_feature('ADX\_BTC', output, prediction\_columns, high, low)

output = talib.APO(features['Close'], fastperiod=12, slowperiod=26, matype=0)

add\_feature('APO\_BTC', output, prediction\_columns, high, low)

output = talib.AROONOSC(features['High'], features['Low'], timeperiod=14)

add\_feature('AROONOSC\_BTC', output, prediction\_columns, high, low)

output = talib.BOP(features['Open'], features['High'], features['Low'], features['Close'])

add\_feature('BOP\_BTC', output, prediction\_columns, high, low)

output = talib.CMO(features['Close'], timeperiod=14)

add\_feature('CMO\_BTC', output, prediction\_columns, high, low)

output = talib.MINUS\_DI(features['High'], features['Low'], features['Close'], timeperiod=14)

add\_feature('MINUS\_DI\_BTC', output, prediction\_columns, high, low)

output = talib.MINUS\_DM(features['High'], features['Low'], timeperiod=14)

add\_feature('MINUS\_DM\_BTC', output, prediction\_columns, high, low)

output = talib.MOM(features['Close'], timeperiod=10)

add\_feature('MOM\_BTC', output, prediction\_columns, high, low)

output = talib.PLUS\_DM(features['High'], features['Low'], timeperiod=14)

add\_feature('PLUS\_DM\_BTC', output, prediction\_columns, high, low)

output = talib.PPO(features['Close'], fastperiod=12, slowperiod=26, matype=0)

add\_feature('PPO\_BTC', output, prediction\_columns, high, low)

output = talib.ROC(features['Close'], timeperiod=10)

add\_feature('ROC\_BTC', output, prediction\_columns, high, low)

output = talib.ROCP(features['Close'], timeperiod=10)

add\_feature('ROCP\_BTC', output, prediction\_columns, high, low)

output = talib.ROCR(features['Close'], timeperiod=10)

add\_feature('ROCR\_BTC', output, prediction\_columns, high, low)

output = talib.ROCR100(features['Close'], timeperiod=10)

add\_feature('ROCR100\_BTC', output, prediction\_columns, high, low)

output = talib.RSI(features['Close'], timeperiod=14)

add\_feature('RSI\_BTC', output, prediction\_columns, high, low)

output = talib.TRIX(features['Close'], timeperiod=30)

add\_feature('TRIX\_BTC', output, prediction\_columns, high, low)

output = talib.MIDPRICE(features['High'], features['Low'], timeperiod=14)

add\_feature('MIDPRICE\_BTC', output, prediction\_columns, high, low)

output = talib.DEMA(features['Close'], timeperiod=30)

add\_feature('DEMA\_BTC', output, prediction\_columns, high, low)

output = talib.EMA(features['Close'], timeperiod=30)

add\_feature('EMA\_BTC', output, prediction\_columns, high, low)

output = talib.HT\_TRENDLINE(features['Close'])

add\_feature('HT\_TRENDLINE\_BTC', output, prediction\_columns, high, low)

output = talib.KAMA(features['Close'], timeperiod=30)

add\_feature('KAMA\_BTC', output, prediction\_columns, high, low)

output = talib.MA(features['Close'], timeperiod=30, matype=0)

add\_feature('MA\_BTC', output, prediction\_columns, high, low)

output = talib.MIDPOINT(features['Close'], timeperiod=14)

add\_feature('MIDPOINT\_BTC', output, prediction\_columns, high, low)

output = talib.SMA(features['Close'], timeperiod=30)

add\_feature('SMA\_BTC', output, prediction\_columns, high, low)

output = talib.TEMA(features['Close'], timeperiod=30)

add\_feature('TEMA\_BTC', output, prediction\_columns, high, low)

output = talib.TRIMA(features['Close'], timeperiod=30)

add\_feature('TRIMA\_BTC', output, prediction\_columns, high, low)

output = talib.WMA(features['Close'], timeperiod=30)

add\_feature('WMA\_BTC', output, prediction\_columns, high, low)

output = talib.ATR(features['High'], features['Low'], features['Close'], timeperiod=14)

add\_feature('ATR\_BTC', output, prediction\_columns, high, low)

output = talib.NATR(features['High'], features['Low'], features['Close'], timeperiod=14)

add\_feature('NATR\_BTC', output, prediction\_columns, high, low)

output = talib.TRANGE(features['High'], features['Low'], features['Close'])

add\_feature('TRANGE\_BTC', output, prediction\_columns, high, low)

output = talib.HT\_DCPERIOD(features['Close'])

add\_feature('HT\_DCPERIOD\_BTC', output, prediction\_columns, high, low)

output = talib.HT\_DCPHASE(features['Close'])

add\_feature('WHT\_DCPHASEMA\_BTC', output, prediction\_columns, high, low)

output = talib.HT\_TRENDMODE(features['Close'])

add\_feature('HT\_TRENDMODE\_BTC', output, prediction\_columns, high, low)

output = talib.BETA(features['High'], features['Low'], timeperiod=5)

add\_feature('BETA\_BTC', output, prediction\_columns, high, low)

output = talib.LINEARREG(features['Close'], timeperiod=14)

add\_feature('LINEARREG\_BTC', output, prediction\_columns, high, low)

output = talib.LINEARREG\_ANGLE(features['Close'], timeperiod=14)

add\_feature('WLINEARREG\_ANGLEMA\_BTC', output, prediction\_columns, high, low)

output = talib.LINEARREG\_INTERCEPT(features['Close'], timeperiod=14)

add\_feature('LINEARREG\_INTERCEPT\_BTC', output, prediction\_columns, high, low)

output = talib.LINEARREG\_SLOPE(features['Close'], timeperiod=14)

add\_feature('LINEARREG\_SLOPE\_BTC', output, prediction\_columns, high, low)

output = talib.STDDEV(features['Close'], timeperiod=5, nbdev=1)

add\_feature('STDDEV\_BTC', output, prediction\_columns, high, low)

output = talib.TSF(features['Close'], timeperiod=14)

add\_feature('TSF\_BTC', output, prediction\_columns, high, low)

output = talib.VAR(features['Close'], timeperiod=5, nbdev=1)

add\_feature('VAR\_BTC', output, prediction\_columns, high, low)

''''''

len(prediction\_columns)-11

#%%

if dop\_fich ==1:

tickerDf\_features = tickerData.history(period = period\_, interval = interval\_)

features = tickerDf\_features[:-1].copy()

##############

high\_fich = add\_all\_ta\_features(features,

open = 'Open',

high = 'High',

low = 'Low',

close = 'Close',

volume = 'Volume',

fillna = True)

low\_fich = add\_all\_ta\_features(features,

open = 'Open',

high = 'High',

low = 'Low',

close = 'Close',

volume = 'Volume',

fillna = True)

for i in high\_fich.columns.tolist()[7:]:

high[i] = high\_fich[i]

low[i] = low\_fich[i]

prediction\_columns.append(i)

############

#60

#Momentum Indicator Functions - 22

output = talib.WILLR(features['High'], features['Low'], features['Close'], timeperiod=14)

add\_feature('WILLR', output, prediction\_columns, high, low)

output = talib.PLUS\_DI(features['High'], features['Low'], features['Close'], timeperiod=14)

add\_feature('PLUS\_DI', output, prediction\_columns, high, low)

output = talib.DX(features['High'], features['Low'], features['Close'], timeperiod=14)

add\_feature('DX', output, prediction\_columns, high, low)

output = talib.CCI(features['High'], features['Low'], features['Close'], timeperiod=14)

add\_feature('CCI', output, prediction\_columns, high, low)

output = talib.ADXR(features['High'], features['Low'], features['Close'], timeperiod=14)

add\_feature('ADXR', output, prediction\_columns, high, low)

output = talib.ULTOSC(features['High'], features['Low'], features['Close'], timeperiod1=7, timeperiod2=14, timeperiod3=28)

add\_feature('ULTOSC', output, prediction\_columns, high, low)

output = talib.ADX(features['High'], features['Low'], features['Close'], timeperiod=14)

add\_feature('ADX', output, prediction\_columns, high, low)

output = talib.APO(features['Close'], fastperiod=12, slowperiod=26, matype=0)

add\_feature('APO', output, prediction\_columns, high, low)

output = talib.AROONOSC(features['High'], features['Low'], timeperiod=14)

add\_feature('AROONOSC', output, prediction\_columns, high, low)

output = talib.BOP(features['Open'], features['High'], features['Low'], features['Close'])

add\_feature('BOP', output, prediction\_columns, high, low)

output = talib.CMO(features['Close'], timeperiod=14)

add\_feature('CMO', output, prediction\_columns, high, low)

output = talib.MINUS\_DI(features['High'], features['Low'], features['Close'], timeperiod=14)

add\_feature('MINUS\_DI', output, prediction\_columns, high, low)

output = talib.MINUS\_DM(features['High'], features['Low'], timeperiod=14)

add\_feature('MINUS\_DM', output, prediction\_columns, high, low)

output = talib.MOM(features['Close'], timeperiod=10)

add\_feature('MOM', output, prediction\_columns, high, low)

output = talib.PLUS\_DM(features['High'], features['Low'], timeperiod=14)

add\_feature('PLUS\_DM', output, prediction\_columns, high, low)

output = talib.PPO(features['Close'], fastperiod=12, slowperiod=26, matype=0)

add\_feature('PPO', output, prediction\_columns, high, low)

output = talib.ROC(features['Close'], timeperiod=10)

add\_feature('ROC', output, prediction\_columns, high, low)

output = talib.ROCP(features['Close'], timeperiod=10)

add\_feature('ROCP', output, prediction\_columns, high, low)

output = talib.ROCR(features['Close'], timeperiod=10)

add\_feature('ROCR', output, prediction\_columns, high, low)

output = talib.ROCR100(features['Close'], timeperiod=10)

add\_feature('ROCR100', output, prediction\_columns, high, low)

output = talib.RSI(features['Close'], timeperiod=14)

add\_feature('RSI', output, prediction\_columns, high, low)

output = talib.TRIX(features['Close'], timeperiod=30)

add\_feature('TRIX', output, prediction\_columns, high, low)

#Overlap Studies Functions - 11

output = talib.MIDPRICE(features['High'], features['Low'], timeperiod=14)

add\_feature('MIDPRICE', output, prediction\_columns, high, low)

output = talib.DEMA(features['Close'], timeperiod=30)

add\_feature('DEMA', output, prediction\_columns, high, low)

output = talib.EMA(features['Close'], timeperiod=30)

add\_feature('EMA', output, prediction\_columns, high, low)

output = talib.HT\_TRENDLINE(features['Close'])

add\_feature('HT\_TRENDLINE', output, prediction\_columns, high, low)

output = talib.KAMA(features['Close'], timeperiod=30)

add\_feature('KAMA', output, prediction\_columns, high, low)

output = talib.MA(features['Close'], timeperiod=30, matype=0)

add\_feature('MA', output, prediction\_columns, high, low)

output = talib.MIDPOINT(features['Close'], timeperiod=14)

add\_feature('MIDPOINT', output, prediction\_columns, high, low)

output = talib.SMA(features['Close'], timeperiod=30)

add\_feature('SMA', output, prediction\_columns, high, low)

output = talib.TEMA(features['Close'], timeperiod=30)

add\_feature('TEMA', output, prediction\_columns, high, low)

output = talib.TRIMA(features['Close'], timeperiod=30)

add\_feature('TRIMA', output, prediction\_columns, high, low)

output = talib.WMA(features['Close'], timeperiod=30)

add\_feature('WMA', output, prediction\_columns, high, low)

#Volatility Indicator Functions - 3

output = talib.ATR(features['High'], features['Low'], features['Close'], timeperiod=14)

add\_feature('ATR', output, prediction\_columns, high, low)

output = talib.NATR(features['High'], features['Low'], features['Close'], timeperiod=14)

add\_feature('NATR', output, prediction\_columns, high, low)

output = talib.TRANGE(features['High'], features['Low'], features['Close'])

add\_feature('TRANGE', output, prediction\_columns, high, low)

#Cycle Indicator Functions - 3

output = talib.HT\_DCPERIOD(features['Close'])

add\_feature('HT\_DCPERIOD', output, prediction\_columns, high, low)

output = talib.HT\_DCPHASE(features['Close'])

add\_feature('WHT\_DCPHASEMA', output, prediction\_columns, high, low)

output = talib.HT\_TRENDMODE(features['Close'])

add\_feature('HT\_TRENDMODE', output, prediction\_columns, high, low)

#Statistic Functions - 8

output = talib.BETA(features['High'], features['Low'], timeperiod=5)

add\_feature('BETA', output, prediction\_columns, high, low)

output = talib.LINEARREG(features['Close'], timeperiod=14)

add\_feature('LINEARREG', output, prediction\_columns, high, low)

output = talib.LINEARREG\_ANGLE(features['Close'], timeperiod=14)

add\_feature('WLINEARREG\_ANGLEMA', output, prediction\_columns, high, low)

output = talib.LINEARREG\_INTERCEPT(features['Close'], timeperiod=14)

add\_feature('LINEARREG\_INTERCEPT', output, prediction\_columns, high, low)

output = talib.LINEARREG\_SLOPE(features['Close'], timeperiod=14)

add\_feature('LINEARREG\_SLOPE', output, prediction\_columns, high, low)

output = talib.STDDEV(features['Close'], timeperiod=5, nbdev=1)

add\_feature('STDDEV', output, prediction\_columns, high, low)

output = talib.TSF(features['Close'], timeperiod=14)

add\_feature('TSF', output, prediction\_columns, high, low)

output = talib.VAR(features['Close'], timeperiod=5, nbdev=1)

add\_feature('VAR', output, prediction\_columns, high, low)

#Math Transform Functions - 13

output = talib.ACOS(features['Close'])

add\_feature('ACOS', output, prediction\_columns, high, low)

output = talib.ASIN(features['Close'])

add\_feature('ASIN', output, prediction\_columns, high, low)

output = talib.ATAN(features['Close'])

add\_feature('ATAN', output, prediction\_columns, high, low)

output = talib.COS(features['Close'])

add\_feature('COS', output, prediction\_columns, high, low)

output = talib.COSH(features['Close'])

add\_feature('COSH', output, prediction\_columns, high, low)

output = talib.EXP(features['Close'])

add\_feature('EXP', output, prediction\_columns, high, low)

output = talib.LN(features['Close'])

add\_feature('LN', output, prediction\_columns, high, low)

output = talib.LOG10(features['Close'])

add\_feature('LOG10', output, prediction\_columns, high, low)

output = talib.SIN(features['Close'])

add\_feature('SIN', output, prediction\_columns, high, low)

output = talib.SINH(features['Close'])

add\_feature('SINH', output, prediction\_columns, high, low)

output = talib.SQRT(features['Close'])

add\_feature('SQRT', output, prediction\_columns, high, low)

output = talib.TAN(features['Close'])

add\_feature('TAN', output, prediction\_columns, high, low)

output = talib.TANH(features['Close'])

add\_feature('TANH', output, prediction\_columns, high, low)

''''''

len(prediction\_columns)-11

#%% md

# Обучение - High

#%%

if diff\_ch == 1:

X = high[prediction\_columns[1:]]

y = high[prediction\_columns[0]]

else:

X = high[prediction\_columns\_high[1:]]

y = high[prediction\_columns\_high[0]]

high\_x\_train, high\_x\_test, high\_train, high\_test = train\_test\_split(X, y, test\_size=0.33, random\_state = 42, shuffle = False)

high\_x\_train.fillna(value = 0, inplace=True)

high\_x\_test.fillna(value = 0, inplace=True)

high\_train.fillna(value = 0, inplace=True)

high\_test.fillna(value = 0, inplace=True)

if choose == 1:

regressor\_high = LinearRegression()

regressor\_high.fit(high\_x\_train, high\_train)

elif choose == 2:

######### RandomForest

max\_depth\_les = 100

min\_samples\_leaf\_les = 10

#########

regressor\_high = RandomForestRegressor(max\_depth=max\_depth\_les, random\_state=0, min\_samples\_leaf=min\_samples\_leaf\_les)

regressor\_high.fit(high\_x\_train, high\_train)

elif choose == 3:

######### CatBoost

iterations\_cat=500 #500

learning\_rate\_cat=0.03 #0.03

depth\_cat=8 #8

l2\_leaf\_reg\_cat=7

#########

regressor\_high = CatBoostRegressor(iterations=iterations\_cat,

learning\_rate=learning\_rate\_cat,

depth=depth\_cat,

l2\_leaf\_reg=l2\_leaf\_reg\_cat)

regressor\_high.fit(high\_x\_train, high\_train)

#%% md

# Feature importance - High

#%%

if choose == 3:

if diff\_ch == 1:

df\_features\_high = pd.DataFrame(regressor\_high.get\_feature\_importance(),

index = prediction\_columns[1:],

columns = ['Feature importance (High)'])

df\_features\_high = df\_features\_high.sort\_values(by=['Feature importance (High)'])

df\_features\_high[-15:].plot(kind='barh', figsize=(10,10))

else:

df\_features\_high = pd.DataFrame(regressor\_high.get\_feature\_importance(),

index = prediction\_columns\_high[1:],

columns = ['Feature importance (High)'])

df\_features\_high = df\_features\_high.sort\_values(by=['Feature importance (High)'])

df\_features\_high[-15:].plot(kind='barh', figsize=(10,10))

#%% md

# Обратные преобразования - High

#%%

if diff\_ch == 1:

inv\_high\_test = inv\_all\_transform(high1, high\_test, lambda\_high, box\_cox\_ch)

inv\_high\_x = inv\_all\_transform(high1, pd.Series(regressor\_high.predict(high\_x\_test)), lambda\_high, box\_cox\_ch)

#%% md

# Обучение - Low

#%%

if diff\_ch == 1:

X\_low = low[prediction\_columns[1:]]

y\_low = low[prediction\_columns[0]]

else:

X\_low = low[prediction\_columns\_low[1:]]

y\_low = low[prediction\_columns\_low[0]]

low\_x\_train, low\_x\_test, low\_train, low\_test = train\_test\_split(X\_low, y\_low, test\_size=0.33, random\_state = 42, shuffle = False)

low\_x\_train.fillna(value = 0, inplace=True)

low\_x\_test.fillna(value = 0, inplace=True)

low\_train.fillna(value = 0, inplace=True)

low\_test.fillna(value = 0, inplace=True)

if choose == 1:

from sklearn.linear\_model import LinearRegression

regressor\_low = LinearRegression()

regressor\_low.fit(low\_x\_train, low\_train)

elif choose == 2:

from sklearn.ensemble import RandomForestRegressor

regressor\_low = RandomForestRegressor(max\_depth=max\_depth\_les, random\_state=0, min\_samples\_leaf=min\_samples\_leaf\_les)

regressor\_low.fit(low\_x\_train, low\_train)

elif choose == 3:

from catboost import CatBoostRegressor

regressor\_low = CatBoostRegressor(iterations=iterations\_cat,

learning\_rate=learning\_rate\_cat,

depth=depth\_cat,

l2\_leaf\_reg=l2\_leaf\_reg\_cat)

regressor\_low.fit(low\_x\_train, low\_train)

#%% md

# Feature importance - Low

#%%

if choose == 3:

if diff\_ch == 1:

df\_features\_low = pd.DataFrame(regressor\_low.get\_feature\_importance(),

index = prediction\_columns[1:],

columns = ['Feature importance (Low)'])

df\_features\_low = df\_features\_low.sort\_values(by=['Feature importance (Low)'])

df\_features\_low[-15:].plot(kind='barh', figsize=(10,10))

else:

df\_features\_low = pd.DataFrame(regressor\_low.get\_feature\_importance(),

index = prediction\_columns\_low[1:],

columns = ['Feature importance (Low)'])

df\_features\_low = df\_features\_low.sort\_values(by=['Feature importance (Low)'])

df\_features\_low[-15:].plot(kind='barh', figsize=(10,10))

#%% md

# Обратные преобразования - Low

#%%

if diff\_ch == 1:

inv\_low\_test = inv\_all\_transform(low1, low\_test, lambda\_low, box\_cox\_ch)

inv\_low\_x = inv\_all\_transform(low1, pd.Series(regressor\_low.predict(low\_x\_test)), lambda\_low, box\_cox\_ch)

#%% md

# Графики - High

#%%

if diff\_ch == 1:

df=pd.DataFrame(inv\_high\_test[-vivod-2:-1])

df.rename(columns = {0:'High'},inplace =True)

df1\_high=pd.DataFrame()

df1\_high['High'] = pd.concat([df['High'], pd.Series(np.NaN)],ignore\_index=True)

df1\_high['Predict\_high'] = np.NaN

df1\_high['Predict\_high'].iloc[-1] = inv\_high\_x[-1]

df1\_high.index = [i for i in range(-vivod,2,1)]

ax = df1\_high[-vivod-2:].plot(figsize=(16,6), linewidth = 1, style='.-', ylabel = 'Цена $', xlabel = 'Часы', xticks = range(-vivod,2,1))

ax.set\_xticklabels(list(range(-vivod,0,1))+['last', 'next'])

else:

df=pd.DataFrame(high\_test.values[-vivod-2:-1])

df.rename(columns = {0:'High'},inplace =True)

df1\_high=pd.DataFrame()

df1\_high['High'] = pd.concat([df['High'], pd.Series(np.NaN)],ignore\_index=True)

df1\_high['Predict\_high'] = np.NaN

df1\_high['Predict\_high'].iloc[-1] = regressor\_high.predict(high\_x\_test)[-1]

df1\_high.index = [i for i in range(-vivod,2,1)]

ax = df1\_high[-vivod-2:].plot(figsize=(16,6), linewidth = 1, style='.-', ylabel = 'Цена $', xlabel = 'Часы', xticks = range(-vivod,2,1))

ax.set\_xticklabels(list(range(-vivod,0,1))+['last', 'next'])

#%% md

# Графики - Low

#%%

if diff\_ch == 1:

df=pd.DataFrame(inv\_low\_test[-vivod-2:-1])

df.rename(columns = {0:'Low'},inplace =True)

df1\_low=pd.DataFrame()

df1\_low['Low'] = pd.concat([df['Low'], pd.Series(np.NaN)],ignore\_index=True)

df1\_low['Predict\_low'] = np.NaN

df1\_low['Predict\_low'].iloc[-1] = inv\_low\_x[-1]

df1\_low.index = [i for i in range(-vivod,2,1)]

ax = df1\_low[-vivod-2:].plot(figsize=(16,6), linewidth = 1, style='.-', ylabel = 'Цена $', xlabel = 'Часы', xticks = range(-vivod,2,1))

ax.set\_xticklabels(list(range(-vivod,0,1))+['last', 'next'])

else:

df=pd.DataFrame(low\_test.values[-vivod-2:-1])

df.rename(columns = {0:'Low'},inplace =True)

df1\_low=pd.DataFrame()

df1\_low['Low'] = pd.concat([df['Low'], pd.Series(np.NaN)],ignore\_index=True)

df1\_low['Predict\_low'] = np.NaN

df1\_low['Predict\_low'].iloc[-1] = regressor\_low.predict(low\_x\_test)[-1]

df1\_low.index = [i for i in range(-vivod,2,1)]

ax = df1\_low[-vivod-2:].plot(figsize=(16,6), linewidth = 1, style='.-', ylabel = 'Цена $', xlabel = 'Часы', xticks = range(-vivod,2,1))

ax.set\_xticklabels(list(range(-vivod,0,1))+['last', 'next'])

#%% md

# Вывод

#%%

df\_for = pd.DataFrame([df1\_high['High'], df1\_high['Predict\_high'], df1\_low['Low'], df1\_low['Predict\_low']]).T

ax = df\_for[-vivod-2:].plot(figsize=(16,6), linewidth = 1, style='.-', ylabel = 'Цена $', xlabel = 'Часы', xticks = range(-vivod,2,1))

ax.set\_xticklabels(list(range(-vivod,0,1))+['last', 'next'])

print('High:', df1\_high['High'].iloc[-2],' -> ', df1\_high['Predict\_high'].iloc[-1], '\nLow: ', df1\_low['Low'].iloc[-2], ' -> ', df1\_low['Predict\_low'].iloc[-1])